PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-062636
(43)Date of publication of application : 07.03.1997

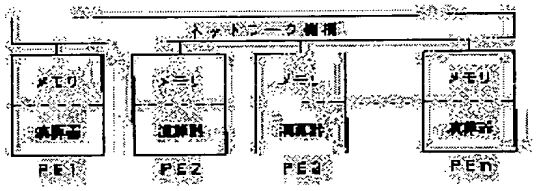
(51)Int.Cl. G06F 15/16
G06F 9/45

(21)Application number : 07-215714
(22)Date of filing : 24.08.1995

(71)Applicant : FUJITSU LTD
(72)Inventor : YAMANAKA EIJI

(54) PROGRAM EXECUTING METHOD AND COMPILING PROCESS METHOD

(57)Abstract:
PROBLEM TO BE SOLVED: To generate an object code for decentralized memory type parallel computers by fully automatic parallelizing by making each processor element perform data processing by using data exploded in its local memory by following a procedure exploded in its local memory.
SOLUTION: A compiler is used to explode all data required for data processing execution in the local memories of respective processor elements PE1-PEn, and divide and explode the procedure required for the data processing execution in the local memories of the processor elements PE1-PEn. Then the respective processor elements PE1-PEn perform the data processing by using the data exploded in their local memories by following the procedure exploded in their local memories. Consequently, the object code for the decentralized memory type parallel computers can be generated by fully automatic parallelizing.



LEGAL STATUS

[Date of request for examination] 29.05.2001
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平9－62636

(43)公開日　平成9年（1997）3月7日

(51)Int.Cl.[6]	識別記号	庁内整理番号	FI		技術表示箇所
G06F 15/16	430		G06F 15/16	430	Z
				430	A
9/45			9/44	322	G

審査請求　未請求　請求項の数7　ＯＬ（全 11 頁）

(21)出願番号　特願平7－215714

(22)出願日　平成7年（1995）8月24日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中4丁目1番1号

(72)発明者　山中　栄次
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

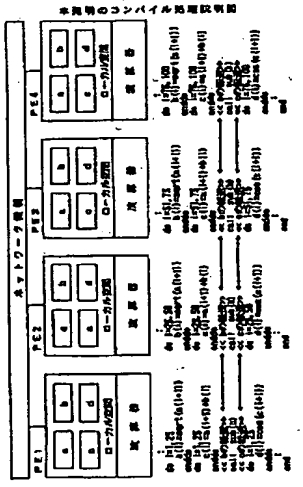
(74)代理人　弁理士　岡田　光由　（外１名）

(54)【発明の名称】プログラム実行方法及びコンパイル処理方法

(57)【要約】

【課題】本発明は、完全自動並列化のコンパイル処理方法の実現を可能にするプログラム実行方法の提供を目的とする。

【解決手段】ローカルメモリを持つ複数のプロセッサエレメントから構成される分散メモリ型並列計算機で実行するプログラム実行方法において、データ処理実行に必要となる全データを、各プロセッサエレメントのローカルメモリに展開し、データ処理実行に必要となる手続きを分割して、各プロセッサエレメントのローカルメモリに展開し、そして、各プロセッサエレメントが、自ローカルメモリに展開される手続きに従い、自ローカルメモリに展開されるデータを使ってデータ処理を実行するように構成する。



1

【特許請求の範囲】

【請求項1】　ローカルメモリを持つ複数のプロセッサエレメントから構成される分散メモリ型並列計算機で実行するプログラム実行方法において、

データ処理実行に必要となる全データを、各プロセッサエレメントのローカルメモリに展開し、

データ処理実行に必要となる手続きを分割して、各プロセッサエレメントのローカルメモリに展開し、

そして、各プロセッサエレメントが、自ローカルメモリに展開される手続きに従い、自ローカルメモリに展開されるデータを使ってデータ処理を実行するよう処理することを、

特徴とするプログラム実行方法。

【請求項2】　請求項1記載のプログラム実行方法において、

各プロセッサエレメントは、自ローカルメモリに展開されるデータのアクセス範囲を限定するよう処理することを、

特徴とするプログラム実行方法。

【請求項3】　請求項1又は2記載のプログラム実行方法において、

各プロセッサエレメントは、手続きの実行途中で、自ローカルメモリに展開されるデータを更新し、そのデータを，そのデータを必要とする他プロセッサエレメントに転送していくよう処理することを、

特徴とするプログラム実行方法。

【請求項4】　ソースプログラムをコンパイルすることで分散メモリ型並列計算機用のオブジェクトコードを生成するコンパイル処理方法において、

手続き部分が並列化できるのか否かを解析して、並列実行可能な手続き部分については、各プロセッサエレメントに担当させる部分を解析する第1の処理過程と、

上記第1の処理過程で解析される各プロセッサエレメントの担当する手続き部分のデータアクセス範囲を解析する第2の処理過程と、

上記第1の処理過程で解析される各プロセッサエレメントの担当する手続き部分に対してデータを共通に割り当てつつ、上記第2の処理過程で解析されるデータアクセス範囲と、プログラム記述とに基づき、プロセッサエレメント間で転送すべきデータの転送形態を解析する第3の処理過程と、

上記第1の処理過程の解析結果と、上記第3の処理過程の解析結果とに基づいてオブジェクトコードを生成する第4の処理過程とを備えることを、

特徴とするコンパイル処理方法。

【請求項5】　請求項4記載のコンパイル処理方法において、

第3の処理過程で、後方で使用されるデータについては、その値が確定した時点から転送開始に入るようにと転送位置を決定するよう処理することを、

2

特徴とするコンパイル処理方法。

【請求項6】　請求項4又は5記載のコンパイル処理方法において、

第3の処理過程で、プロセッサエレメントにより個別更新される配列部分については、その配列部分を他の全てのプロセッサエレメントに転送するようにと転送先を決定するよう処理することを、

特徴とするコンパイル処理方法。

【請求項7】　請求項4又は5記載のコンパイル処理方法において、

第3の処理過程で、特定のプロセッサエレメントのみが必要とするデータについては、そのデータをそのプロセッサエレメントにのみ転送するようにと転送先を決定するよう処理することを、

特徴とするコンパイル処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、分散メモリ型並列計算機用のオブジェクトコードを生成するコンパイル処理方法と、分散メモリ型並列計算機で実行するプログラム実行方法とに関し、特に、完全自動並列化を実現するコンパイル処理方法と、完全自動並列化のコンパイル処理方法の実現を可能にするプログラム実行方法とに関する。

【0002】

【従来の技術】分散メモリ型並列計算機は、図10に示すように、メモリを有する演算器（プロセッサエレメント／ＰＥ）がネットワーク機構で複数結合された形式のアーキテクチャを持つ計算機である。この分散メモリ型並列計算機では、自ＰＥ上のデータについては、通常のアクセス方式でアクセスすることが可能であるが、他のＰＥ上のデータについては、ネットワーク機構を経由してアクセスしなければならない。

【0003】このようなメモリアーキテクチャを対象にコンパイラで自動並列化を行う場合、データをＰＥ上に非常にうまく分散して配置する必要があるが、現在のコンパイラの自動並列化機能では、手続き部分については自動並列化を実現できるものの、データの自動分散配置を行うことは非常に難しいというのが実情である。

【0004】この問題点を解決するために、ＰＥ間で共有される仮想的なメモリ空間を、各ＰＥのメモリとネットワーク機構を用いて仮想的に構成することが行われている。この仮想的なメモリ空間は仮想グローバル空間と呼ばれ、各ＰＥ毎に分かれた通常のメモリ空間はローカル空間と呼ばれている。図11に、このメモリアーキテクチャを図示する。

【0005】この仮想グローバル空間は、全てのＰＥからアクセスすることが可能であるが、ネットワーク機構を介するため、アクセス速度が遅い。これに対して、ローカル空間は、自ＰＥからだけアクセスすることが可能

であり、他ＰＥからアクセスすることは不可能であるが、アクセス速度は速い。

【０００６】これから、コンパイラの自動並列化において、データの自動分散配置の問題点を回避するために、仮想グローバル空間を使用する構成を採ると、データへのアクセス速度が遅くなり、性能が著しく低下するという問題点が出ることになる。一方、性能を重視して、ローカル空間を使用すると、自動並列化において困難なデータの自動分散配置を実現する技術を構築しなければならないという問題点がある。

【０００７】そこで、従来では、自動並列化を行うために、ユーザに対してデータの分散配置形態を指示させて、この指示に基づいてデータを分散配置する構成を採って、自動並列化機能で、なるべくローカル空間を使用するようにしつつ、ローカル空間を使用できない部分だけ、仮想グローバル空間を使用するようにして並列化を行うという構成を採っている。

【０００８】

【発明が解決しようとする課題】しかしながら、このような従来技術に従っていると、ユーザがデータの分散配置を指示しなければならないことから、完全な自動並列化になっていないという問題点があった。そして、プログラムの内容によって仮想グローバル空間を使用せざるを得ない場合があり、当たり外れが激しく、コンスタントな性能向上を期待できないという問題点があった。そして、データを分散配置することから、並列化の対象となるループが、回転間のデータ依存関係の無いものに限定されてしまうという問題点もあった。

【０００９】本発明はかかる事情に鑑みてなされたものであって、分散メモリ型並列計算機用のオブジェクトコードを完全自動並列化でもって生成する新たなコンパイル処理方法の提供と、完全自動並列化のコンパイル処理方法の実現を可能にする新たなプログラム実行方法の提供とを目的とする。

【００１０】

【課題を解決するための手段】図１に本発明の原理構成を図示する。図中、１はデータ処理装置であって、本発明を実現するコンパイラ１０を展開するものである。

【００１１】このコンパイラ１０は、ソースプログラムをコンパイルすることで分散メモリ型並列計算機用のオブジェクトコードを生成するものであって、ソースプログラムを字句の列に変換して、その字句列からソースプログラムの文法的な構造を再構成し、更に意味解析を実行する字句・構文解析部１１と、字句・構文解析部１１の解析結果を使って、オブジェクトコードを生成するコード生成部１２と、字句・構文解析部１１とコード生成部１２との間に位置して、コード生成の最適化を行う最適化部１３と、全体の制御処理を実行する制御部１４とを備える。

【００１２】この最適化部１３は、本発明を実現するために、手続き部分が並列化できるのか否かを解析して、並列実行可能な手続き部分については、各プロセッサエレメントに担当させる部分を解析する第１の解析部１５と、第１の解析部１５で解析される各プロセッサエレメントの担当する手続き部分のデータアクセス範囲を解析する第２の解析部１６と、第１の解析部１５で解析される各プロセッサエレメントの担当する手続き部分に対してデータを共通に割り当てつつ、第２の解析部１６で解析されるデータアクセス範囲と、プログラム記述とに基づき、プロセッサエレメント間で転送すべきデータの転送形態を解析する第３の解析部１７とを備える。

【００１３】そして、コード生成部１２は、第１の解析部１５の解析結果と、第３の解析部１７の解析結果とに基づいてオブジェクトコードを生成していくように処理する。

【００１４】このように構成される本発明では、第１の解析部１５は、ソースプログラムの持つ手続き部分のみを処理対象として、それらの手続き部分が並列化できるのか否かを解析して、並列実行可能な手続き部分については、各プロセッサエレメントに担当させる部分を解析する。すなわち、ソースプログラムの持つデータの配列については並列化対象としない。これから、各プロセッサエレメントは、オブジェクトコードがローディングされると、データについては、ソースプログラムに記述される配列宣言文に従って、データファイルからその全てをコピーして持つことになる。

【００１５】一方、第２の解析部１６は、第１の解析部１５で解析される各プロセッサエレメントの担当する手続き部分のデータアクセス範囲を解析し、これを受けて、第３の解析部１７は、各プロセッサエレメントの担当する手続き部分に対してデータを共通に割り当てつつ、第２の解析部１６で解析されるデータアクセス範囲と、プログラム記述とに基づき、プロセッサエレメント間で転送すべきデータの転送形態を解析する。

【００１６】例えば、第３の解析部１７は、プロセッサエレメントにより個別更新される配列部分については、その配列部分を他の全てのプロセッサエレメントに転送するようにと転送先を決定することで、ローディング後のデータの整合性を保つように処理する。また、後方で使用されるデータについては、その値が確定した時点から転送開始に入るようにと転送位置を決定することで、データ転送処理と演算処理とを重ね合わせてデータ転送処理を隠蔽するように処理する。また、特定のプロセッサエレメントのみが必要とするデータについては、そのデータをそのプロセッサエレメントにのみ転送するようにと転送先を決定することで、転送コストの削減を図るように処理するのである。

【００１７】このように、本発明のコンパイラ１０では、分散メモリ型並列計算機でデータ処理を実行する場合に、手続き部分のみを自動的に並列化し、データにつ

いては、分散配置させずに全プロセッサエレメントがそのコピーを持つ構成を採って、必要に応じてデータを一致させつつ、必要なデータをやり取りする構成を採ることで、完全自動並列化を実現するものである。

【００１８】そして、この本発明のコンパイラ１０を用いることで、データ処理実行に必要となる全データを、各プロセッサエレメントのローカルメモリに展開し、データ処理実行に必要となる手続きを分割して、各プロセッサエレメントのローカルメモリに展開し、そして、各プロセッサエレメントが、自ローカルメモリに展開される手続きに従い、自ローカルメモリに展開されるデータを使ってデータ処理を実行するという分散メモリ型並列計算機上での新たなプログラム実行方法を実現できるようになる。

【００１９】この新たなプログラム実行方法では、各プロセッサエレメントは、自ローカルメモリに展開されるデータのアクセス範囲を限定するよう処理する。そして、各プロセッサエレメントは、手続きの実行途中で、自ローカルメモリに展開されるデータを更新し、そのデータを、そのデータを必要とする他プロセッサエレメントに転送していくよう処理することになる。

【００２０】なお、この新たなプログラム実行方法は、本発明のコンパイラ１０を用いることで実現できるものであるが、本発明のコンパイラ１０を用いなくても実現可能である。

【００２１】

【発明の実施の形態】以下、実施の形態に従って本発明を詳細に説明する。図２に、本発明のコンパイラ１０が実行する最適化処理の処理構成を図示する。

【００２２】この図に示すように、本発明のコンパイラ１０は、最適化処理に入ると、先ず最初に、手続き部並列化解析処理を実行する。この手続き部並列化解析処理では、ソースプログラムの持つ手続き部（主にＤＯループである）が並列化できるのか否かを解析して、並列化できるＤＯループについては、ループ分割情報を生成する。すなわち、本発明のコンパイラ１０では、ソースプログラムの持つデータについては分割しない構成を採るのである。

【００２３】この手続き部並列化解析処理を終了すると、続いて、データアクセス範囲解析処理を実行する。このデータアクセス範囲解析処理では、手続き部並列化解析処理で求められた並列化できるＤＯループのデータアクセス範囲を解析して、アクセス範囲情報を生成する。

【００２４】このデータアクセス範囲解析処理を終了すると、続いて、最適データ転送解析処理を実行する。この最適データ転送解析処理では、データアクセス範囲解析処理で生成されたアクセス範囲情報と、プログラム記述とに基づき、プロセッサエレメント間で転送すべきデータの転送情報を生成してから、その転送情報を最適化することで最適なデータ転送形態を決定する。

【００２５】図３に、この最適化処理の更に詳細な処理フローを図示する。すなわち、本発明のコンパイラ１０は、最適化処理に入ると、手続き毎に、この処理フローに示すように、手続き部並列化解析処理に入って、先ず最初に、ソースプログラムに記述されるデータフローを解析し、続いて、ソースプログラムに記述される制御フローを解析し、続いて、ＤＯループ毎に、配列アクセスの添字からデータ依存関係を解析し、続いて、これらの解析結果に基づき、ＤＯループが並列実行可能か否かを解析して、並列実行可能なＤＯループについてはループ分割情報を生成する。

【００２６】そして、手続き部並列化解析処理を終了すると、データアクセス範囲解析処理に入って、並列実行可能なＤＯループ毎に、配列アクセスの添字からアクセス範囲を解析して、そのアクセス範囲情報を生成する。

【００２７】そして、データアクセス範囲解析処理を終了すると、最適データ転送解析処理に入って、先ず最初に、ＤＯループ毎に、生成したループ分割情報とアクセス範囲情報とに基づき必要とされるデータ転送を解析して、転送対象／転送位置／転送先を指定するデータ転送情報を生成し、続いて、その生成したデータ転送情報を最適化する。

【００２８】この最適化処理を実行すると、本発明のコンパイラ１０は、生成したループ分割情報に従って分割されるＤＯループを持つオブジェクトコードを生成するとともに、生成した最適化データ転送情報に従って作成されるデータ転送命令及びデータ受信命令を持つオブジェクトコードを生成する。

【００２９】次に、具体例に従って、本発明のコンパイラ１０が実行する最適化処理について説明する。図４に、ソースプログラムの一例を図示する。

【００３０】このソースプログラムは、１０１個のデータ要素を持つ配列ａと、１００個のデータ要素を持つ配列ｂと、１０１個のデータ要素を持つ配列ｃと、１００個のデータ要素を持つ配列ｄとを定義して、先ず最初に、

ｂ（ｉ）＝ｓｑｒｔ（ａ（ｉ＋１））　　ｉ＝１～１００

に従って、データ要素ａ（ｉ＋１）の平方根で定義されるデータ要素ｂ（ｉ）を求めることで配列ｂを算出し、続いて、

ｃ（ｉ）＝ａ（ｉ＋１）＋ｂ（ｉ）　　ｉ＝１～１００

に従ってデータ要素ｃ（ｉ）を求めることで配列ｃを算出し、続いて、配列ｂを用いるサブルーチン「ｓｕｂ（ｂ）」を呼び出した後、それに続いて、

ｄ（ｉ）＝ｃｏｓ（ｃ（ｉ＋１））　　ｉ＝１～１００

に従って、データ要素ｃ（ｉ＋１）のコサイン関数値で

定義されるデータ要素ｄ（ｉ）を求めることで配列ｄを算出していくプログラムである。

【００３１】このようなソースプログラムをコンパイルする場合、分散メモリ型並列計算機が４台のプロセッサエレメントで構成されるときには、本発明のコンパイラ１０は、最適化処理に入ると、配列ａ／配列ｂ／配列ｃ／配列ｄについては分割しないようにしながら、配列ｂを求めるＤＯループと、配列ｃを求めるＤＯループと、配列ｄを求めるＤＯループについては、並列化が可能であることを解析して、それらのＤＯループを４台のプロセッサエレメントで並列処理すべく、「ｉ＝１～25」、「ｉ＝26～50」、「ｉ＝51～75」、「ｉ＝76～100 」という添字のグループに分割する。

【００３２】続いて、「ｉ＝１～25」を実行するプロセッサエレメントは、配列ｂの算出にあたって、自エレメントに展開される配列ａのデータ要素ａ(2) ～ａ(26)を使用し、配列ｃの算出にあたって、自エレメントに展開される配列ａのデータ要素ａ(2) ～ａ(26)と、自エレメントで算出した配列ｂのデータ要素ｂ(1) ～(25)とを使用し、配列ｄの算出にあたって、自エレメントで算出した配列ｃのデータ要素ｃ(2) ～(25)と、隣接プロセッサエレメントの算出した配列ｃのデータ要素ｃ(26)とを使用するということを解析する。

【００３３】また、「ｉ＝26～50」を実行するプロセッサエレメントは、配列ｂの算出にあたって、自エレメントに展開される配列ａのデータ要素ａ(27)～ａ(51)を使用し、配列ｃの算出にあたって、自エレメントに展開される配列ａのデータ要素ａ(27)～ａ(51)と、自エレメントで算出した配列ｂのデータ要素ｂ(26)～(50)とを使用し、配列ｄの算出にあたって、自エレメントで算出した配列ｃのデータ要素ｃ(27)～(50)と、隣接プロセッサエレメントの算出した配列ｃのデータ要素ｃ(51)とを使用するということを解析する。

【００３４】また、「ｉ＝51～75」を実行するプロセッサエレメントは、配列ｂの算出にあたって、自エレメントに展開される配列ａのデータ要素ａ(52)～ａ(76)を使用し、配列ｃの算出にあたって、自エレメントに展開される配列ａのデータ要素ａ(52)～ａ(76)と、自エレメントで算出した配列ｂのデータ要素ｂ(51)～(75)とを使用し、配列ｄの算出にあたって、自エレメントで算出した配列ｃのデータ要素ｃ(52)～(75)と、隣接プロセッサエレメントの算出した配列ｃのデータ要素ｃ(76)とを使用するということを解析する。

【００３５】また、「ｉ＝76～100 」を実行するプロセッサエレメントは、配列ｂの算出にあたって、自エレメントに展開される配列ａのデータ要素ａ(77)～ａ(101)を使用し、配列ｃの算出にあたって、自エレメントに展開される配列ａのデータ要素ａ(77)～ａ(101)と、自エレメントで算出した配列ｂのデータ要素ｂ(76)～(100)とを使用し、配列ｄの算出にあたって、自エレメントで算出した配列ｃのデータ要素ｃ(76)～(100)と、自エレメントに展開される配列ｃのデータ要素ｃ(101)の初期値とを使用するということを解析する。

【００３６】そして、サブルーチン「ｓｕｂ（ｂ）」の実行前に、各プロセッサエレメントが配列ｂを持つ必要があることを解析して、その「ｓｕｂ（ｂ）」の前までに、各プロセッサエレメントで生成された配列ｂの配列部分を、他の全てのプロセッサエレメントに転送することを指示するデータ転送情報を生成する。なお、サブルーチンの他、ソースプログラムに入出力命令が記述されているときや、多重ＤＯループが記述されているときなどには、このように、各プロセッサエレメントで生成された配列部分を他の全てのプロセッサエレメントに転送していく必要が起こる。

【００３７】更に、配列ｄの算出前に、「ｉ＝１～25」を実行するプロセッサエレメントが、「ｉ＝26～50」を実行するプロセッサエレメントからデータ要素ｃ(26)を受け取る必要があり、「ｉ＝26～50」を実行するプロセッサエレメントが、「ｉ＝51～75」を実行するプロセッサエレメントからデータ要素ｃ(51)を受け取る必要があり、「ｉ＝51～75」を実行するプロセッサエレメントが、「ｉ＝76～100 」を実行するプロセッサエレメントからデータ要素ｃ(76)を受け取る必要があることを解析して、その配列ｄの算出時点に、「ｉ＝26～50」を実行するプロセッサエレメントの持つデータ要素ｃ(26)を、「ｉ＝１～25」を実行するプロセッサエレメントに転送し、「ｉ＝51～75」を実行するプロセッサエレメントの持つデータ要素ｃ(51)を、「ｉ＝26～50」を実行するプロセッサエレメントに転送し、「ｉ＝76～100 」を実行するプロセッサエレメントの持つデータ要素ｃ(76)を、「ｉ＝51～75」を実行するプロセッサエレメントに転送することを指示するデータ転送情報を生成する。

【００３８】本発明のコンパイラ１０が、ここで最適化処理を終了して、データ転送情報についてはこれ以上の最適化処理を実行しないときには、この最適化処理に従って生成されるオブジェクトコードが４台のプロセッサエレメントにローディングされることで、各プロセッサエレメントは、図５に示すようなデータ処理を実行する。

【００３９】すなわち、ＰＥ１ないしＰＥ４で示される４台のプロセッサエレメントは、配列ａ／配列ｂ／配列ｃ／配列ｄのデータ要素の実体を図示しないデータファイルから読み込むことで、ソースプログラムの記述する配列宣言文の指すデータのコピーを展開する。

【００４０】そして、コンパイラ１０の最適化処理で生成されたＤＯループの分割情報により、ＰＥ１のプロセッサエレメントは、並列実行処理に従いつつ、

ｂ（ｉ）＝ｓｑｒｔ（ａ（ｉ＋１））　　ｉ＝１～25

に従って、データ要素ｂ（ｉ）を求め、

c（i）＝a（i＋1）＋b（i）　　i＝1～25

に従ってデータ要素c（i）を求め、

d（i）＝cos（c（i＋1））　　i＝1～25

に従ってデータ要素d（i）を求める。

【0041】また、コンパイラ10の最適化処理で生成されたDOループの分割情報により、PE2のプロセッサエレメントは、並列実行処理に従いつつ、

b（i）＝sqrt（a（i＋1））　　i＝26～50

に従って、データ要素b（i）を求め、

c（i）＝a（i＋1）＋b（i）　　i＝26～50

に従ってデータ要素c（i）を求め、

d（i）＝cos（c（i＋1））　　i＝26～50

に従ってデータ要素d（i）を求める。

【0042】また、コンパイラ10の最適化処理で生成されたDOループの分割情報により、PE3のプロセッサエレメントは、並列実行処理に従いつつ、

b（i）＝sqrt（a（i＋1））　　i＝51～75

に従って、データ要素b（i）を求め、

c（i）＝a（i＋1）＋b（i）　　i＝51～75

に従ってデータ要素c（i）を求め、

d（i）＝cos（c（i＋1））　　i＝51～75

に従ってデータ要素d（i）を求める。

【0043】また、コンパイラ10の最適化処理で生成されたDOループの分割情報により、PE4のプロセッサエレメントは、並列実行処理に従いつつ、

b（i）＝sqrt（a（i＋1））　　i＝76～100

に従って、データ要素b（i）を求め、

c（i）＝a（i＋1）＋b（i）　　i＝76～100

に従ってデータ要素c（i）を求め、

d（i）＝cos（c（i＋1））　　i＝76～100

に従ってデータ要素d（i）を求める。

【0044】このとき、コンパイラ10の最適化処理で生成されたデータ転送情報により、PE1のプロセッサエレメントは、図6上段に示すように、サブルーチン「sub（b）」の実行に入る前に、自エレメントで算出した配列bのデータ要素b(1)～b(25)を、PE2／PE3／PE4のプロセッサエレメントに転送し、PE2のプロセッサエレメントは、図6下段に示すように、サブルーチン「sub（b）」の実行に入る前に、自エレメントで算出した配列bのデータ要素b(26)～b(50)を、PE1／PE3／PE4のプロセッサエレメントに転送し、PE3のプロセッサエレメントは、図7上段に示すように、サブルーチン「sub（b）」の実行に入る前に、自エレメントで算出した配列bのデータ要素b(51)～b(75)を、PE1／PE2／PE4のプロセッサエレメントに転送し、PE4のプロセッサエレメントは、図7下段に示すように、サブルーチン「sub（b）」の実行に入る前に、自エレメントで算出した配列bのデータ要素b(76)～b(100)を、PE1／PE2／PE3のプロセッサエレメントに転送していく。

【0045】そして、コンパイラ10の最適化処理で生成されたデータ転送情報により、PE1のプロセッサエレメントは、PE2／PE3／PE4のプロセッサエレメントから、配列bのデータ要素b(26)～b(50)と、配列bのデータ要素b(51)～b(75)と、配列bのデータ要素b(76)～b(100)とを受け取ることを確認すると、サブルーチン「sub（b）」の実行に入り、PE2のプロセッサエレメントは、PE1／PE3／PE4のプロセッサエレメントから、配列bのデータ要素b(1)～b(25)と、配列bのデータ要素b(51)～b(75)と、配列bのデータ要素b(76)～b(100)とを受け取ることを確認すると、サブルーチン「sub（b）」の実行に入り、PE3のプロセッサエレメントは、PE1／PE2／PE4のプロセッサエレメントから、配列bのデータ要素b(1)～b(25)と、配列bのデータ要素b(26)～b(50)と、配列bのデータ要素b(76)～b(100)とを受け取ることを確認すると、サブルーチン「sub（b）」の実行に入り、PE4のプロセッサエレメントは、PE1／PE2／PE3のプロセッサエレメントから、配列bのデータ要素b(1)～b(25)と、配列bのデータ要素b(26)～b(50)と、配列bのデータ要素b(51)～b(75)とを受け取ることを確認すると、サブルーチン「sub（b）」の実行に入っていく。

【0046】そして、コンパイラ10の最適化処理で生成されたデータ転送情報により、PE2のプロセッサエレメントは、サブルーチン「sub（b）」の実行が終了すると、図8上段に示すように、PE1のプロセッサエレメントが配列dの算出に入る前に、その算出で必要となる配列cのデータ要素c(26)を転送し、PE4のプロセッサエレメントは、サブルーチン「sub（b）」の実行が終了すると、図8上段に示すように、PE3のプロセッサエレメントが配列dの算出に入る前に、その算出で必要となる配列cのデータ要素c(76)を転送し、PE3のプロセッサエレメントは、サブルーチン「sub（b）」の実行が終了すると、図8下段に示すように、PE2のプロセッサエレメントが配列dの算出に入る前に、その算出で必要となる配列cのデータ要素c(51)を転送していく。

【0047】このようにして、本発明のコンパイラ10

では、分散メモリ型並列計算機において、手続き部分のみを自動的に並列化し、データについては、分散配置させずに全プロセッサエレメントがそのコピーを持つ構成を採って、必要に応じてデータを一致させつつ、必要なデータをやり取りする構成を採ることで、完全自動並列化を実現するものである。

【００４８】この実施例では、配列ｄの算出で必要となる配列ｃの境界部分のデータ要素については、そのデータ要素のみをプロセッサエレメント間で転送していくという最適化されたデータ転送方法を用いる構成を開示した。この構成に従うと転送コストの削減を図ることができるが、このような最適化を行わずに、配列ｃの全データ要素を転送対象とするデータ転送方法を採ることも可能である。

【００４９】また、分散メモリ型並列計算機がデータ転送処理と演算処理とを並列処理できる機能を持つ場合には、後方で使用されるデータについては、その値が確定した時点から転送開始に入るようにと転送位置を最適化することで、データ転送処理と演算処理とを重ね合わせてデータ転送処理を隠蔽することが可能である。

【００５０】例えば、図４のソースプログラムをコンパイルする場合、図９に示すように、サブルーチン「ｓｕｂ（ｂ）」の実行時点で、その演算に必要となる配列ｂのデータ要素の転送を実行するのではなくて、配列ｂのデータ要素の値が確定した時点から転送処理に入るようにし、また、配列ｄの算出時点で、その算出に必要となる配列ｃのデータ要素の転送を実行するのではなくて、その配列ｃのデータ要素の値が確定した時点から転送処理に入るようにすることで、データ転送処理を隠蔽するのである。

【００５１】この本発明のコンパイラ１０を用いることで、データ処理実行に必要となる全データを、各プロセッサエレメントのローカルメモリに展開し、データ処理実行に必要となる手続きを分割して、各プロセッサエレメントのローカルメモリに展開し、そして、各プロセッサエレメントが、自ローカルメモリに展開される手続きに従い、自ローカルメモリに展開されるデータを使ってデータ処理を実行するという分散メモリ型並列計算機上での新たなプログラム実行方法を実現できるようになる。

【００５２】この新たなプログラム実行方法では、各プロセッサエレメントは、自ローカルメモリに展開されるデータのアクセス範囲を限定するよう処理する。そして、各プロセッサエレメントは、手続きの実行途中で、自ローカルメモリに展開されるデータを更新し、そのデータを、そのデータを必要とする他プロセッサエレメントに転送していくよう処理することになる。

【００５３】なお、この新たなプログラム実行方法は、本発明のコンパイラ１０を用いることで実現できるものであるが、本発明のコンパイラ１０を用いなくても実現可能である。

【００５４】

【発明の効果】以上説明したように、本発明のコンパイラでは、分散メモリ型並列計算機でデータ処理を実行する場合に、手続き部分のみを自動的に並列化し、データについては、分散配置させずに全プロセッサエレメントがそのコピーを持つ構成を採って、必要に応じてデータを一致させつつ、必要なデータをやり取りする構成を採る。

【００５５】これにより、完全自動並列化を実現できるようになる。そして、プログラマのノウハウによらないで完全自動並列化を実現できるようになることから、当たり外れがなくなり、コンスタントな性能向上を期待できるようになる。そして、データを分散配置することから、並列化の対象となるループの範囲が拡大できるようになる。

【００５６】そして、本発明のコンパイラ等を用いることで、本発明のプログラム実行方法が実現できるようになり、逆に言えば、本発明のプログラム実行方法により、完全自動並列化を実現する本発明のコンパイラが構築できるようになる。

【図面の簡単な説明】

【図１】本発明の原理構成図である。

【図２】本発明のコンパイラが実行する最適化処理の処理構成図である。

【図３】最適処理の詳細な処理フローである。

【図４】ソースプログラムの一例である。

【図５】本発明のコンパイル処理説明図である。

【図６】本発明のコンパイル処理説明図である。

【図７】本発明のコンパイル処理説明図である。

【図８】本発明のコンパイル処理説明図である。

【図９】本発明のコンパイル処理説明図である。

【図１０】分散メモリ型並列計算機の説明図である。

【図１１】メモリアーキテクチャの説明図である。

【符号の説明】

１　データ処理装置

１０　コンパイラ

１１　字句・構文解析部

１２　コード生成部

１３　最適化部

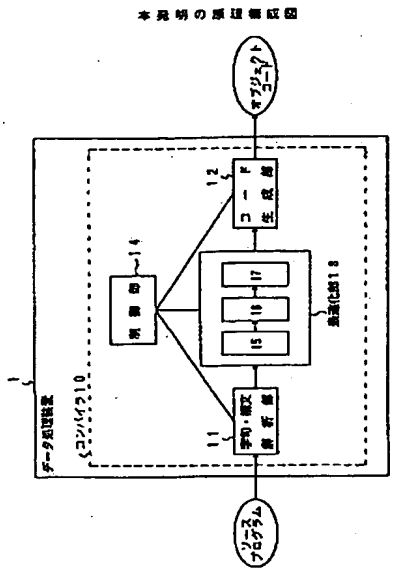
１４　制御部

１５　第１の解析部

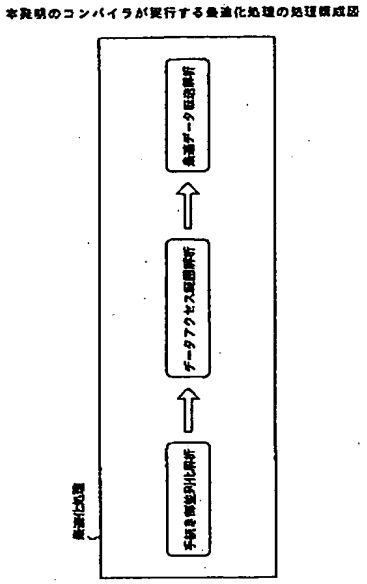
１６　第２の解析部

１７　第３の解析部

【図1】



【図2】

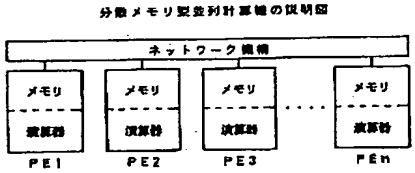


【図4】

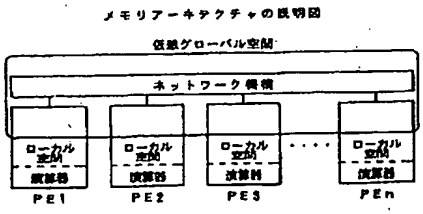
ソースプログラムの一例

```
program sample
real*8 a(101), b(100), c(101), d(100)
   :
do i=1, 100
   b(i)=sqrt(a(i+1))
enddo
do i=1, 100
   c(i)=a(i+1)+b(i)
enddo
call sub(b)
do i=1, 100
   d(i)=cos(c(i+1))
enddo
  :
end
```

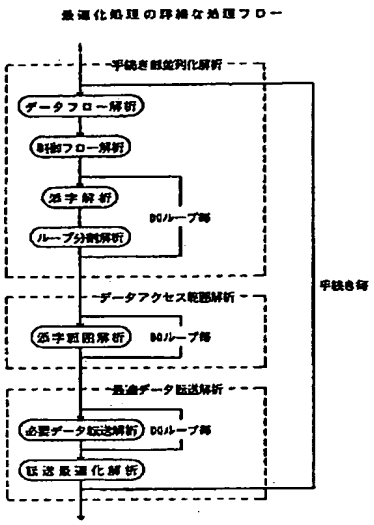
【図10】



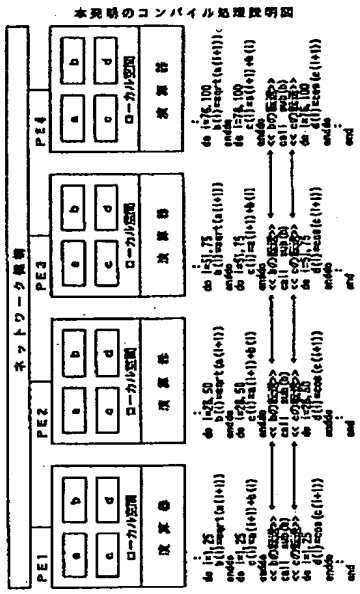
【図11】



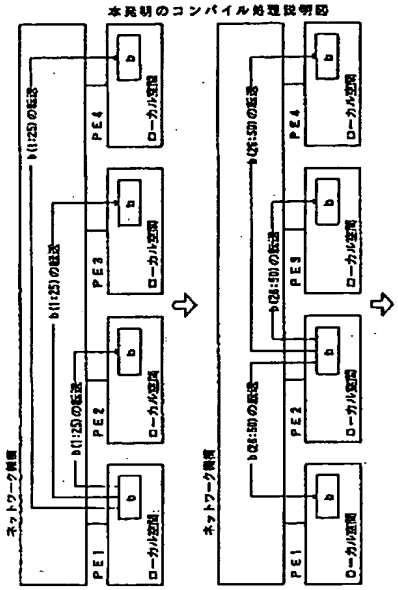
【図3】


最適化処理の詳細な処理フロー
手続き間並列化解析
データフロー解析
制御フロー解析
添字解析
DOループ等
ループ分割解析
データアクセス範囲解析
添字範囲解析
DOループ等
最適データ転送解析
必要データ転送解析
DOループ等
転送最適化解析
手続き毎

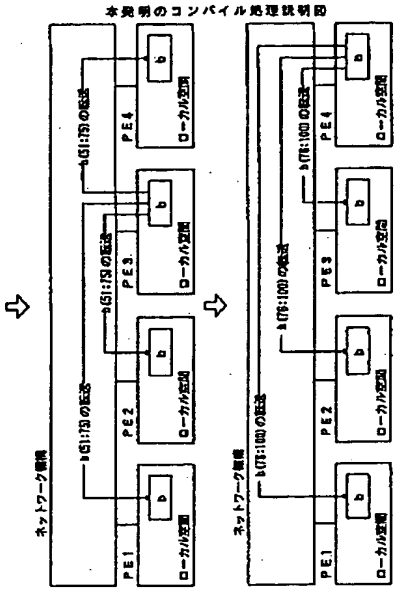
【図5】


本発明のコンパイル処理説明図
ネットワーク装置
PE1
PE2
PE3
PE4
ローカル空間
演算部

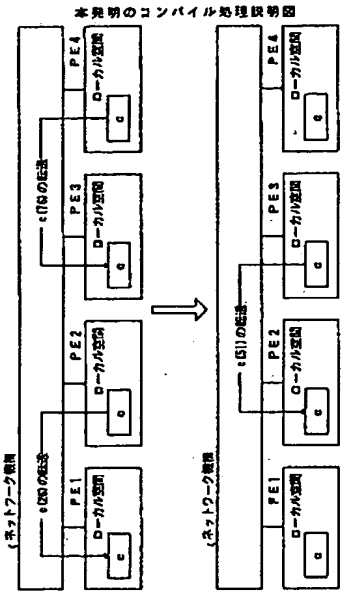
【図6】


本発明のコンパイル処理説明図
PE4
PE3
PE2
PE1
ローカル空間
ネットワーク領域

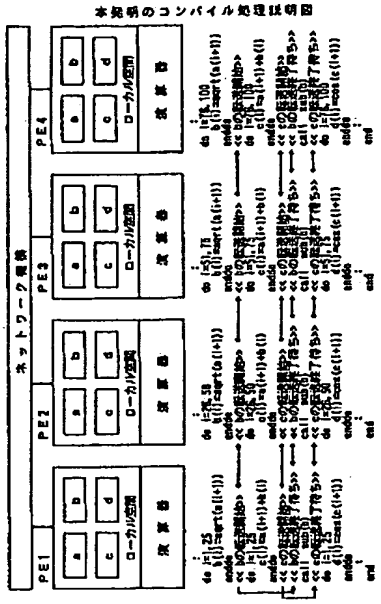
【図7】


本発明のコンパイル処理説明図
PE4
PE3
PE2
PE1
ローカル空間
ネットワーク領域

【図8】


本発明のコンパイル処理説明図
ネットワーク結合
PE1
PE2
PE3
PE4
ローカル空間
a
c
d
e
c(1)の転送
c(0)の転送
c(1)の転送

【図9】


本発明のコンパイル処理説明図
ネットワーク結合
PE1
PE2
PE3
PE4
ローカル空間
a
b
c
d
処理部